Generation of the Ford Sequence of Length 2^n, n Large

H. Fredricksen
Communications Systems Research Section

This article presents three algorithms for forming the Ford sequence of length 2^n and compares the storage requirements for each of the three. These sequences are used in checkout of digital communications equipment.

I. Introduction

Shift register sequences have had application in code generation, prescribed period sequence generation for countdown circuits, and PN shift register sequences have been used for recovering signals from noise in deep space transmissions.

A special sequence period for a shift register of n stages is the deBruijn sequence of length 2^n. In the deBruijn sequence all 2^n possible n-tuples occur once as n successive bits of the cyclic shift of the sequence of length 2^n. These sequences have been used in forming comma-free codes of higher index, as random bit generators when all 2^n possible subsequences of length n are required, and as a test sequence to map through all 2^n states of a Viterbi convolutional decoder.

The method most often used to find a deBruijn sequence of length 2^n is to find a primitive polynomial of degree n over GF[2]. When the primitive polynomial is wired into a shift register, a sequence of length $2^n - 1$ is formed, if care is taken to avoid the all-zero cycle of length 1. An extra logical expression is then required to "add" the zero sequence into the PN sequence to form the deBruijn sequence of length 2^n.

There are $\phi(2^n - 1)/n$ primitive polynomials of degree n. But since there are 2^{2^n-1-n} deBruijn sequences of length 2^n, we see the "linear" deBruijn sequences form a vanishingly small fraction of all deBruijn sequences. Also to find a primitive polynomial of high degree is not necessarily an easy task.

Unfortunately, to generate a nonlinear deBruijn sequence is not generally easy either. There is an algorithm, which we attribute to Ford (Ref. 1) which yields a nonlinear deBruijn sequence. In Ref. 2 the Ford algorithm is investigated and the positions of the truth table for its generation are determined. The algorithms for the Ford sequence generation are given below.

However, to form the Ford sequence using Ford's original algorithm or the algorithm for the truth table requires 2^n bits of storage in the first case, or $(n-1) \times (Z(n)-1)$ bits of storage in the second, where $Z(n)-1$ is the number of positions which are equal to 1 in the truth table generation and $Z(n)$ is given by

$$Z(n) = \frac{1}{n} \sum_{d/n} \phi(d)\, 2^{n/d}$$

We give a new algorithm below which yields the Ford sequence and requires no storage beyond two holding registers of length n bits. The algorithm is valid even for very large n.

II. Algorithms for Generation of Ford Sequence

Ford's Algorithm. Let $x_0 = x_1 = \cdots = x_{n-1} = 0$. The x_{n+k}th bit is a 1 if the n-tuple $x_{k+1}x_{k+2} \cdots x_{k+n-1}\ 1$ has not occurred previously in the sequence, otherwise it is a 0.

Proof of Ford's Algorithm. The process must terminate at $1000 \cdots 0$ for if it terminates at $y_0, y_1, \cdots, y_{n-1} \neq 10 \cdots 0$, then $y_0 \cdots y_{n-1}$ must have occurred at least twice in the sequence, which is not permitted. Also every n-tuple must be on the sequence for if $z_0 \cdots z_{n-1}$ is not on the sequence then neither is one of its possible successors, in particular $z_1 \cdots z_{n-1}\ 0$. Continuing we see $z_2 \cdots z_{n-1}\ 0\ 0$ is not on the sequence, and finally we find that $100 \cdots 0$ is not on the sequence.

We now present the algorithm which determines the truth table for the Ford sequence.

Algorithm 1

(1) Form the pure cycle decomposition of the deBruijn graph, i.e., choose all cycles of length ℓ, $\ell \mid n$.

(2) For each cycle (excepting (0)), find the maximum element, $m_i = 2^{r_i}k_i$, k_i odd, $r_i \geqq 0$.

(3) $\alpha_i = (k_i - 1)/2$.

Algorithm 1 yields $Z(n) - 1$ positions α_i which are the positions which are 1 in the truth table, where $Z(n)$ is the number of cycles of length ℓ, $\ell \mid n$.

Verification of Algorithm 1 is given in Ref. 2. Ford's algorithm requires the whole sequence be saved for the generation and Algorithm 1 requires the saving of the positions $0, y_1, \cdots, y_{n-1}$ which will take the 1 successor $y_1, \cdots, y_{n-1}, 1$.

We now give an algorithm to produce the Ford sequence for large n. The algorithm is similar to Algorithm 1. Algorithm 2 will produce the next n-tuple of the Ford sequence from the current n-tuple.

Algorithm 2

(1) $\beta_0 = (0, 0, 0, \cdots, 0)$, the starting n-tuple of all zeros. (From $\beta_i = (b_1, b_2, \cdots, b_n)$, we produce $\beta_{i+1} = (b_2, b_3, \cdots, b_{n+1})$).

(2) Form $\beta_i^* = (b_2, b_3, \cdots, b_n, 1)$.

(3) Consider all cyclic shifts of β_i^* to find the maximum element M_i on the cycle

$$\beta_i^*,\ M_i = (b_i \cdots b_n, 1; b_2 \cdots b_{i-1})$$

(4) If $b_2 = b_3 = \cdots = b_{i-1} = 0$, then

$$\beta_{i+1} = (b_2, b_3, \cdots, b_n, \overline{b}_1)$$

otherwise $\beta_{i+1} = (b_2, \cdots, b_n, b_1)$.

Proof

Algorithm 2 follows easily from Algorithm 1. If

$$b_2 = b_3 = \cdots = b_{i-1} = 0$$

then the maximum element on one of the pure cycles in Algorithm 1 is

$$m_i = 2^{i-2}\left[1 + \sum_{j=1}^{n-i+1} b_{n-j+1}\,2^j\right]$$

and

$$1 + \sum_{j=1}^{n-i+1} b_{n-j+1}\,2^j = k_i \text{ of Algorithm 1}$$

$$\sum_{j=0}^{n-i} b_{n-j+1}\,2^j = \alpha_i \text{ of Algorithm 1}$$

Algorithm 2 requires saving only the present state β_i and the current largest value of the shift of the vector β_i^*.

References

1. Ford, L. R., Jr., *A Cyclic Arrangement of M-tuples*, Report No. P-1071. Rand Corp., Santa Monica, Calif., Apr. 23, 1957.

2. Fredricksen, H., "The Lexicographically Least deBruijn Cycle," *Journal of Combinatorial Theory*, Vol. 9, No. 1, pp. 1–5, July 1970.